

Université de Rennes 1
IFSIC
Master 1 d'informatique
Programmation, logique et calcul

1ère session 2004-2005
7 janvier 2005, 14-16 h

Ce sujet comporte 3 pages.

Tous documents autorisés.

Le barème est indicatif.

Toutes les réponses doivent être justifiées (de façon concise mais précise).

Partie I - Arbres de recherche (8 points)

Soit le programme

$s(L, V, 1) :- p(L, V), !.$
 $s(L, V, 2) :- q(L, V).$

$p([], 0).$
 $p([_X, _X|L], N) :- !, p(L, M), N \text{ is } M+1.$
 $p([X, Y|L], N) :- b(X, Y), !, p(L, M), N \text{ is } M+2.$
 $p([_X, _Y|L], N) :- p(L, N).$

$q([_X], 0).$
 $q([X, _Y|L], N) :- e(X), q(L, M), N \text{ is } M+1.$
 $q([_X, _Y|L], N) :- q(L, N).$

$b(X, _Y) :- c(X), !.$
 $b(X, Y) :- d(X, Y).$

$c(aa).$ $c(ee).$

$d(cc, dd).$ $d(ee, ff).$

$e(aa).$

Question 1

Dessiner l'arbre de recherche du but suivant : $s([aa, bb, cc, cc, ee], R, S).$

et indiquer les témoins des preuves si il y en a. Indiquer clairement pour chaque coupure les branches qu'elle coupe.

Partie II - Programmation logique (12 points)

On souhaite enrichir le système de planification automatique étudié en TP6 en prenant en compte les effectifs des sessions et les capacités des salles. Les données sont supposées fournies sous le format utilisé en TP sauf pour la description des salles qui devra indiquer leur capacité et un numéro unique dans un prédicat `salle/3` de type

`salle(NuméroDeSalle : entier, NomDeSalle : chaîne de car, Capacité : entier).`

L'effectif de chaque session sera déduit du nombre d'inscrits à chaque session.

Pour simplifier la programmation, on suppose que tous les calculs faits par le TP6 constituent une première phase, terminée par l'énumération (*labeling*), et que tous les traitements liés à la prise en compte des capacités et effectifs constituent une seconde phase, exécutée après l'énumération. Cela signifie que la seconde phase peut toujours s'appuyer sur une solution de la première, et qu'en particulier une allocation de session à des plages horaires est connue. L'objectif de la seconde phase est de vérifier que cette allocation est compatible avec les contraintes liées aux capacités et effectifs.

En résumé, après la première phase, on dispose de deux listes, appelons-les `LvarPlageSession` et `LnomSession`, qui contiennent respectivement les numéros de plages allouées et les noms de sessions. La seconde phase demande de créer une nouvelle liste, `LvarSalleSession`, qui contiendra les numéros de salles allouées aux sessions.

Les contraintes d'effectif et capacité sont de deux types :

C1. Toute session doit avoir lieu dans une salle de capacité supérieure ou égale à l'effectif de la session.

C2. Il n'y a pas plus d'une session par salle et par plage de temps.

L'organisation des calculs suit les points suivants :

1. Une liste de noms de session et une liste de variables de session sont établies. Une variable d'un rang r représente la salle allouée, mais encore inconnue, à la session de rang r .
2. Une liste des effectifs de chaque session, rangés dans l'ordre des sessions, est établie à partir de la base de donnée des inscrits.
3. Pour chaque effectif de session, une liste des numéros de salles de capacité suffisante est établie. Cette liste constitue le domaine de la variable de même rang que la session. Cela permet d'exprimer la contrainte C1.
4. Une liste de paires p (numéro de plage, variable de salle) est établie à partir de la liste de numéros de plage obtenue dans la première phase, celle du TP, et la liste de variables de salle. On doit n'apparier que des numéros et des variables de même rang.
5. La liste de paires p (numéro de plage, variable de salle) est triée selon les numéros de plage.
6. La liste de paires triée est partitionnée en une liste de listes de paires de numéros de plage égaux. On observe que dans cette liste de listes de paires, les variables de salle d'une même liste de paires correspondent à toutes les sessions qui ont lieu dans la même plage temporelle. Toutes ces variables doivent être différentes pour satisfaire la contrainte C2.
7. Après une seconde énumération, les résultats de l'allocation des salles peuvent être affichés.

Question 2 (1 point)

Programmer le point 1 en réutilisant le plus possible les prédicats fournis dans le TP6.

Pour l'ensemble des questions qui suivent, on peut définir autant de prédicats intermédiaires que jugé nécessaire, et on peut aussi réutiliser des prédicats définis dans le TP6.

Question 3 (4 points)

Programmer un prédicat qui réalise le point 3 en supposant le point 2 réalisé. Le prédicat aura la spécification suivante :

```
% recup_salles_compatibles_et_pose_C1( +Leffectif, +LvarSalleSession )
```

Utiliser un prédicat `domaine/2` qui prend en paramètre une variable et une liste de valeurs possibles et assigne l'ensemble de ces valeurs comme domaine à la variable.

Question 4 (2 points)

Programmer un prédicat qui réalise le point 4. Le prédicat aura la spécification suivante :

```
% zip( +Liste1, +Liste2, -Lpaire )  
% ex. zip( [1,2, 2], [a,b, c], [p(1, a), p(2, b), p(2, c)] ) .
```

Question 5 (4 points)

Programmer deux prédicats qui réalisent le point 6 en supposant que le point 5 est réalisé. Les prédicats auront les spécifications suivantes :

```
% partition( +Lpaire, -LLpaire )  
% ex. partition( [p(1, a), p(2, b), p(2, c)], [ [p(1, a)], [p(2, b), p(2, c)] ] )  
% pose_C2( +LLpaire )
```

Question 6 (1 point)

L'organisation en deux phases proposée cause la répétition inutile de certains calculs. Lesquels ?